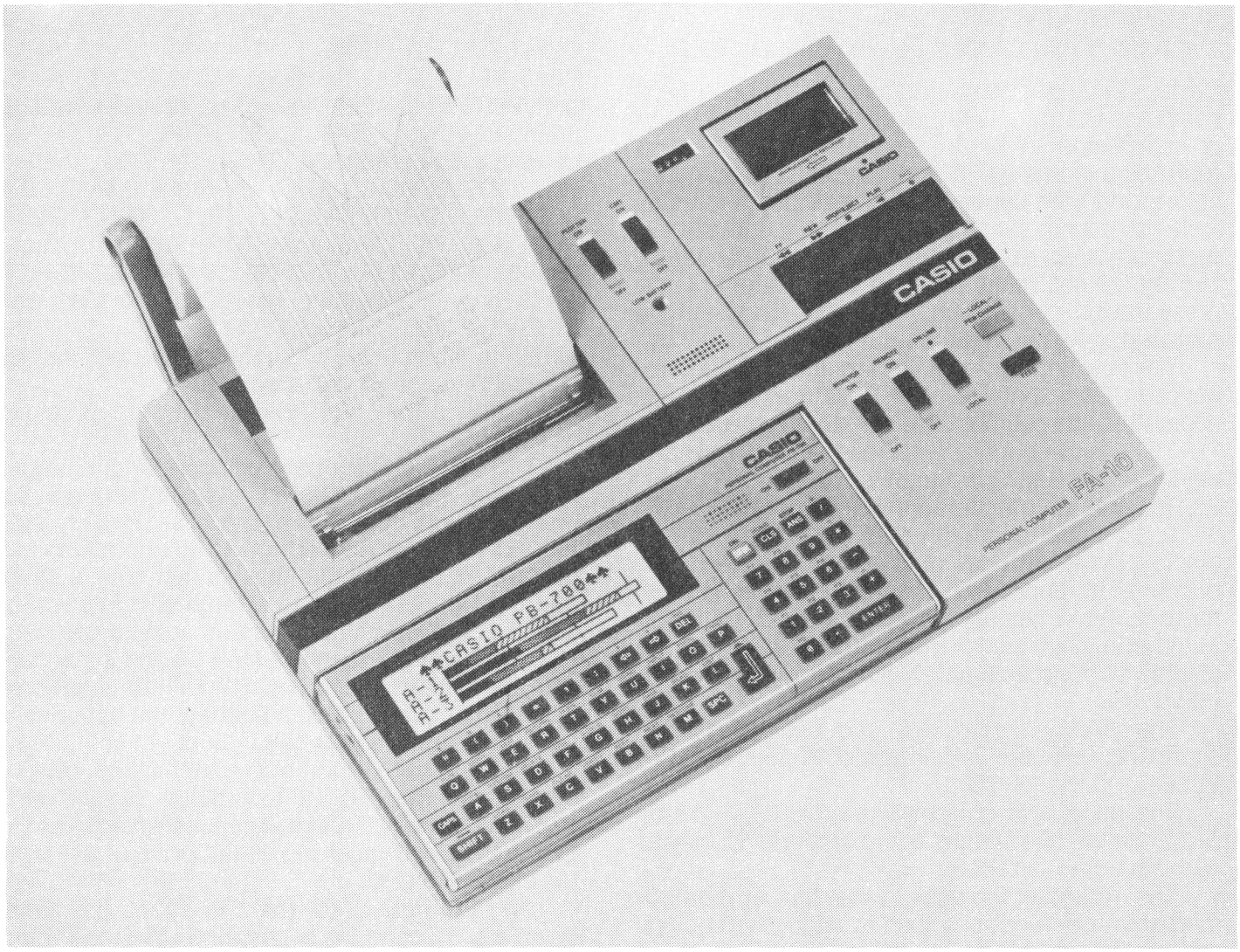
POCKET COMPUTER NEWSLETTER

© Copyright 1984 POCKET COMPUTER NEWSLETTER

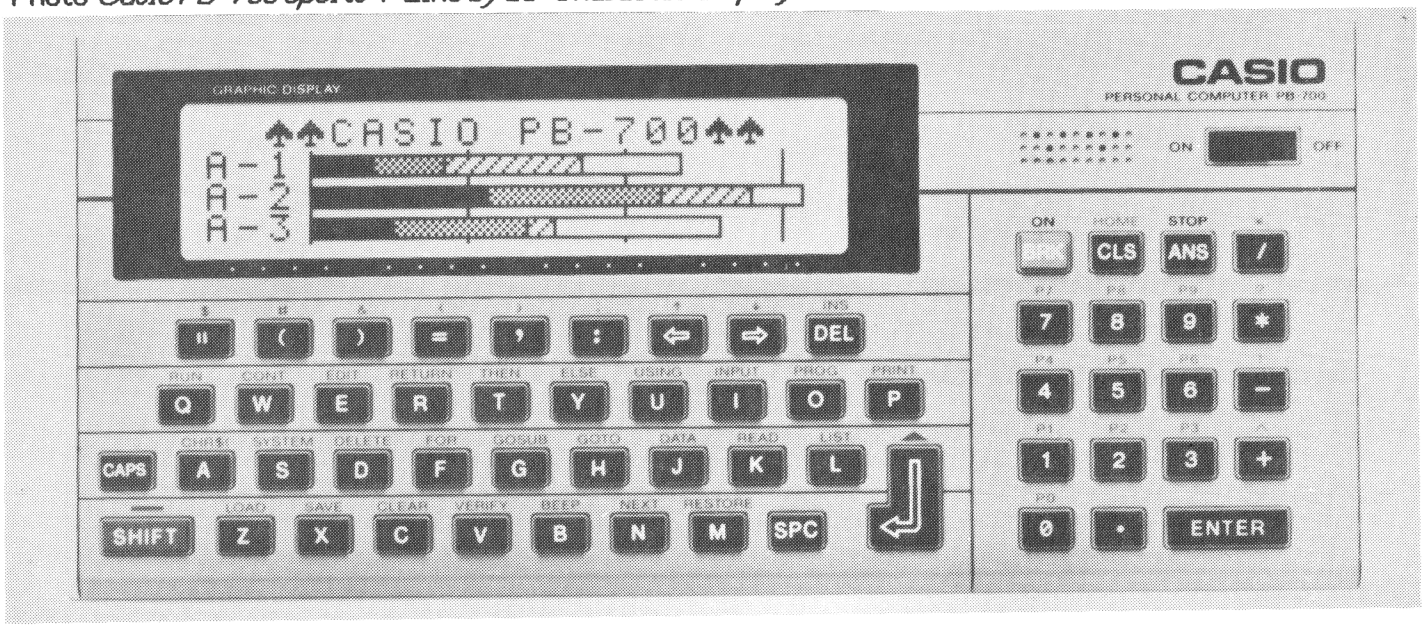
Issue 33 - May/June

Photo *Casio PB-700 Mounted on FA-10 Equipped with CM-1 Micro Cassette Module.*



Another *Personal Information*™ product.

Photo *Casio PB-700 Sports 4-Line by 20-Character Display.*



CASIO INTRODUCES PB-700 HAND-HELD

Casio, Inc., recently introduced a hand-held computer that sports 4K of user memory. Memory is expandable in 4K increments to 16K. It also features a 4-line by 20-character display capable of printing both upper- and lower-case letters as well as graphic symbols. Alternately, the display can be addressed on a pixel-by-pixel basis over its 160 by 32 dot range, making it suitable for the presentation of graphs, charts, and simple pictorials.

The new unit has a 58 keys including a numeric keypad. Commonly used BASIC keywords are accessible through a shifted key feature. The PB-700 has full editing features that facilitate the keying in of a BASIC program.

The unit includes an audio annunciator.

It runs for approximately 100 hours (actual display time) on a set of 4 AA cells. User RAM memory is backed-up by a separate lithium battery. Main battery power is conserved by an automatic shutdown feature that turns off the display after approximately 8 minutes of computer inactivity.

The BASIC interpreter and general system operating software is contained in 25K of ROM installed in the PB-700.

The hand-held unit measure 7/8 by 7-7/8 by 3-1/2 inches and weighs approximately 11 ounces with batteries installed.

The machine includes a number of built-in mathematical functions such as trigonometric and reverse-trigonometric, logarithmic, powers, exponentials, square roots, rounding, random numbers and others. Its command set includes: CONT, DELETE, EDIT, LIST, LLIST, LOAD, NEW, PASS, PROG, RUN, SAVE, SYSTEM TROFF, TRON and VERIFY. BASIC keywords include: ANGLE, BEEP, CHAIN, CLEAR, CLS, DATA, DIM, DRAW, DRAWC, END, ERASE, FOR-TO-STEP, NEXT, GET, GOSUB, GOTO, IF-THEN-ELSE, INPUT, LET, LOCATE, LPRINT, PRINT, PUT, READ, REM, RESTORE, RETURN and STOP. String functions include: ASC, CHR$, LEFT$, LEN, MID$, RIGHT$, STR$ and VAL. Other functions provided are: INKEY$, POINT, TAB and USING.

Up to 10 separate programs or routines may be stored in memory simultaneously.

The list price of the PB-700 in the U.S. is said to be $199.95.

The PB-700 can be expanded into a larger system by using the Casio FA-10 plotter/cassette interface unit. The FA-10 has a 4-1/2 inch wide four color printer/plotter. Up to 80 characters can be printed on a single line. It can hold a small supply of paper or utilize a large external roll of paper. A cassette interface allows data transfers to take place between the PB-700 and a standard external cassette recorder. The FA-10 is powered by internal rechargeable batteries. The list price in the U.S. is stated as $249.95.

An optional micro cassette module can be mounted on the FA-10 to provide integral data storage/recovery. This module is referred to as the CM-1 and has a suggested retail price of $89.95 in the United States.

An alternative to the FA-10 is the FA-4 interface. It contains a cassette interface and a Centronics port enabling the unit to send data to a variety of external printers. The FA-4 interface lists at a U.S. price of $69.95.

HP-71B PRODUCT REVIEW

Hewlett-Packard has finally produced a computer that will fit in a pocket. Well, let's say a good-sized pocket, such as a jacket pocket. The unit is close in size and weight to a Sharp PC-1500 or Radio Shack PC-2. Press releases incorrectly reported that the unit was just one-half an inch thick. That is simply not true. It is about one inch in thickness at the back, sloping down to 3/4-inch at the front.

The unit is powered by four AAA cells which will supposedly operate the unit for approximately 60 hours of "on" time or several months of normal usage. Alternately, the machine may be powered by an a.c. adaptor (sold separately).

In essence, the HP-71B is a stripped down and compacted HP-75. In its unadorned state it has just about the same capabilities of the 75 sans the magnetic strip card reader, the HP-IL interface, and a few software facilities, such as the 75's appointment/alarm scheduler. The strip card reader and HP-IL interface can be added as extra cost options. With its built-in clock and calendar functions, you can program your own appointment and alarm utilities.

However, it appears that HP learned some things from experience with the earlier HP-75. For instance, a rather disconcerting feature of the 75 was the fact that you had to hold down its shift key while you punched the desired "shifted" key. On the HP-71B this nuisance has been removed. Once tapped, a shift key determines the mode of the next key to be pressed. There are two shift keys on the 71B to facilitate packing the keyboard with a lot of easily-accessed features.

The keyboard is arranged in QWERTY format. It is too small for touch-typing, hence the keys are in the traditional HP calculator style, with sloping fronts. Some of the keys have a secondary legend on the sloping front portion. A third set of legends is shown directly above the keys. The second and third legends are accessed via the two shift keys (colored blue and yellow to correspond with the key legends). The keyboard feel is virtually identical to that of a HP-41 series calculator.

The display is about four inches in length. The central portion of the display, consisting of a dot matrix, is about 3 inches in length and 1/4-inch high. It is flanked at each end by a number of special annunciators. The dot matrix can present up to 22 characters (with scrolling across a line of up to 96 characters). Alternately, it may be operated in a graphics mode. The display intensity (reflectivity) may be adjusted through a software command.

Software Powerhouse

The HP-71B actually packs more software power than the considerably larger 75. It comes equipped with 64K of ROM (versus the 75's 48K) and 17.5K of RAM. About 1K of RAM is used by the operating system, but this still leaves a healthy 16+ kilobytes for the user in a stock model. (Currently, RAM can be expanded to 32K by adding 4K modules which are available at extra cost.)

Within the 64K of ROM is a powerful version of HP BASIC containing some 240 statements, commands and functions. Perhaps the most useful feature of the 71B is its powerful file-handling capabilities. All programs and data bases are accessed as independent, memory-resident files. You can have countless files of various types in memory all at one time. A series of programs can all access a common data file. Programs can be chained together. Even better, one program can be "called" (and recursively at that) by another. There is complete control over variables (including the deletion of individual variables or arrays). These are powerful capabilities and a real joy to have available on such a physically small unit. (How many of us wish such refinements were available on desktop-sized computers?)

To give you an idea of the power of the HP-71B, here is a list of the keywords available to the user along with brief comments on some of the more unusual or particularly powerful ones: ABS, ACOS, ADD (used by the built-in statistics package), ADDR$ (returns hexadecimal address of a file), ADJABS, ADJUST and AF (these three keywords are used to adjust the system's clock), AND, ANGLE (polar), ASIN, ASSIGN #, ATAN, AUTO, BEEP and BEEP OFF/ON, BYE (power-down), CALL (true call to another subprogram where variables can be local or global), CAT, CAT$, CEIL (ceiling), CFLAG (clears user or system flags), CHAIN, CHARSET (used to define alternate character set), CHARSET$, CHR$, CLAIM PORT (used to reclaim RAM memory that had been assigned to particular RAM modules), CLASS, CLSTAT (clears statistical array), CONT, CONTRAST (display intensity), COPY (files), CORR (correlation function used in statistics package), COS, CREATE, DATA, DATE (current calendar date), DATE$, DEF FN (define single- or multiple-line user-defined functions), DEF KEY, DEFAULT EXTEND/ON/OFF (sets math exception traps), DEG, DEGREES, DELAY (sets line and character scroll rates), DELETE (lines), DESTROY (delete variable(s) and arrays from memory), DIM, DISP (display on the LCD screen), DISP USING, DISP$, DIV (integer quotient division),

DROP (a pair of coordinates in a data set used by the statistical package), DTH$ (convert decimal to hexadecimal), DVZ (status of divide-by-zero flag), EDIT, END, END DEF/SUB (terminate function definition or subprogram), ENDLINE, ENG (set engineering display format), EPS (epsilon), ERRL, ERRM$, ERRN, EXACT (calibrate system clock), EXOR, EXP, EXPM-1 (natural antilog minus 1), EXPONENT, FACT (factorial), FETCH (a line within current file), FETCH KEY, FIX (fixed format), FLAG (returns condition of a specified flag), FLOOR, FN (executes user-defined function), FOR...NEXT, FP (fractional part), FREE PORT (transfers RAM in a module from main memory to independent use), GDISP (graphic display), GDISP$, GOSUB, GOTO, HTD (hexadecimal to decimal), IF...THEN...ELSE, IMAGE, INF, INPUT, INT, INTEGER, INX (inexact flag status), IP (integer part), IVL (invalid flag status), KEY$, KEYDEF$, KEYDOWN (indicates whether a key is being pressed), LC (upper/lower case toggle), LEN, LET, LINPUT (input an entire line at a time), LIST, LOCK, LOG (natural logarithm), LOGP1 (logarithm of argument plus 1), LOG10, LR (linear regression), MAX, MAXREAL, MEAN, MEM, MERGE (part or all of a file), MIN, MINREAL, MOD (modulo), NAME (a file), NAN (not-a-number function), NOT, NUM, OFF (turns off the unit), OFF ERROR/TIMER (deactivates previous ON ERROR or ON TIMER # statement), ON ERROR GOSUB/GOTO (error trap), ON...GOSUB/GOTO/RESTORE, ON TIMER # (used to enable any of three separate timers), OPTION ANGLE/BASE/ROUND, OR, OVF (overflow flag), PAUSE (suspends program execution), PEEK$, PI, PLIST, POKE, POP (subroutine stack to cancel next return instruction), POS (substring position), PREDV (predicted value, used as a statistical function), PRINT, PRINT USING, PRINT #, PRIVATE (provides file protection), PROTECT (a magnetic card track), PURGE (files), PUT (character input), PWIDTH (defines line length), RAD, RADIANS, RANDOMIZE, READ, READ #, REAL, RED (reduction to return a remainder), REM, RENAME (a file), RENUMBER (any part or all of a program file), RES (result), RESET/RESET CLOCK (resets system flags and clock adjustment factor), RESTORE, RESTORE #, RETURN, RMD (returns the remainder), RND (generates a random number), RUN, SCI (scientific notation), SDEV (standard deviation), SECURE (protects a file from alteration or purging), SETDATE, SETTIME, SFLAG (set flag), SGN, SHORT (defines 5-digit precision), SHOW PORT (displays type and size of all plug-in memory devices), SIN, SQR, STARTUP (command string that is executed when unit is turned on), STAT, STD, STOP, STR$, SUB (denotes beginning of a subprogram), TAN, TIME, TIME$, TOTAL (of a particular variable in a statistical array), TRACE FLOW/VARS/OFF (debugging aid), TRANSFORM (text files to program files and vice versa), TRAP, UNF, UNPROTECT (remove write-protection from a magnetic card track), UNSECURE (a file), UPRC$ (convert lower case letters to upper case), USER (activate/deactivate user-defined keys), VAL, VER$, WAIT, WIDTH, and WINDOW.

As you may infer from the instruction set, this is a rich and powerful implementation with many subtleties. For example, while the PEEK$ function will allow you to examine memory (by nibbles, since the HP-71B is a nibble-oriented machine), it does not allow you to examine a file that has been designated a "private" file. The built-in statistical functions will be of value to many mathematicians and analysists and save a great deal of work in that area. People with special needs can create their own symbol sets and redefine the keyboard to reflect their requirements. The file handling capabilities are superb. In summary, the 71B sports an instruction set that will likely please even hardened programming professionals.

In fact, the instruction set is so rich that you have to be careful not to trip yourself up. For instance, you can control the length of time a message is displayed on the unit's screen through use of the DELAY statement. This delay can range from fractions of a second to infinity. Of course, when you use a delay of infinity, you are really opting to have the message stay on the screen until the user presses the END LINE key. Now the HP-71B has another feature called a "type-ahead" keyboard buffer that allows you to enter information before an input prompt even appears on the screen. Guess what happens if you use a DISPlay statement with a DELAY of infinity and later use an input statement? It appears that the END LINE input used to terminate the DISPlay gets placed in the "type-ahead" keyboard buffer. When the following input statement is processed, it finds that same END LINE character already in the keyboard buffer and assumes that it means the end of the input. Ooops! One way to get around this situation is to execute a statement such as setting a dummy variable equal to the function KEY$ just before executing the INPUT directive. The KEY$ function clears out the keyboard buffer, thus knocking out the troublesome END LINE character.

Add-Ons

Two optional devices that fit within the case of the HP-71B provide it with the capability to rival the HP-75. The first such add-on is for transferring information to and from magnetic strip cards. The

second is an HP-IL (Hewlett-Packard Interface Loop) adaptor.

The magnetic strip card device allows the unit to use magnetic strip-cards. These cards are about 10 inches long and 3/8-inches in width. Each card contains two 650-byte tracks. Processing both sides of a card takes in the order of 15 seconds. It is a convenient way of storing relatively small programs and blocks of data.

Adding the HP-IL adaptor allows the unit to communicate with a variety of peripheral devices ranging from video drivers, to several types of printers and plotters, a mass storage digital tape drive, and other devices such as measuring and test instruments. It is a relatively simple matter, using such peripherals, to have the 71 serve as a data logger, etc. The HP-IL driver software is provided as part of the system software.

The HP-71B is definitely a first-class product. If you have professional requirements and can afford its top-of-the-line price (about $500) you will likely find it an instrument that provides a great deal of a special kind of satisfaction: the satisfaction that comes from owning a product that exhibits far above average quality and capability.

REVIEW OF ROM SOFTWARE MODULES

The following reviews and commentaries were provided by: *Robert Sutliff, 55 North Avenue, New Rochelle, NY 10805.* Readers are advised that the opinions stated herein are those of Mr. Sutliff and do not necessarily reflect the views of *PCN* or its editorial staff.

Introduction

At the end of 1983, Sharp Corporation released its long awaited line of Software Libraries for the PC-1500 and/or PC-1500A. Each module is the size of a CE-155 RAM module and occupies the expansion port in the back of the computer. Only one module can be installed at a time. Denoted the CE-500 series, each module adds 16K of ROM memory, at hexadecimal addresses 0000H through 3FFFh. Although the ending address of actual content for each module is different, the remaining space cannot be used since it is ROM. The Reserve memory space is controlled by the module, so that in the PC-1500 the usual Reserve area is left free for RAM, thereby giving a total of 2047 bytes of user memory in addresses 4000H through 47FFH.

Each module contains programs and subroutines, written in BASIC by American Micro Products, Inc., which are addressed by softkeys. Each module comes with a detailed manual providing operating instructions and examples. Modules can be found through mail order houses and dealers ranging at anywhere from $25 to $60 each, so a buyer should shop around! The main advantage of a module is that 16K of programs are loaded by a quick installation, as opposed to the lengthy cassette loading procedure. Access time to any program in the module is instantaneous. The programs in any module (except SHARPCALC) can be (L)LISTed in the usual fashion, although no lines can be altered. The ROM programs act as though they have been MERGEd into current memory. That is, new lines in RAM appear after the ROM programs when LISTed. Programs in RAM must be labeled and run by using the label and DEF or GOTO. The CE-150 printer must be used for certain modules; for others, it is an optional device and printer related prompts will not appear if it is unattached. The modules use the Reserve keys for program selection and provide extensive prompts, default values, and recovery-from-error traps. In particular, default values often flash and make it easy to rerun a problem by changing only certain parameters. Some of the modules are excellent in covering their subject matter, while others appear deficient and contain programming errors. The following discussion reviews several of the modules.

Graphics Development (CE-501A)

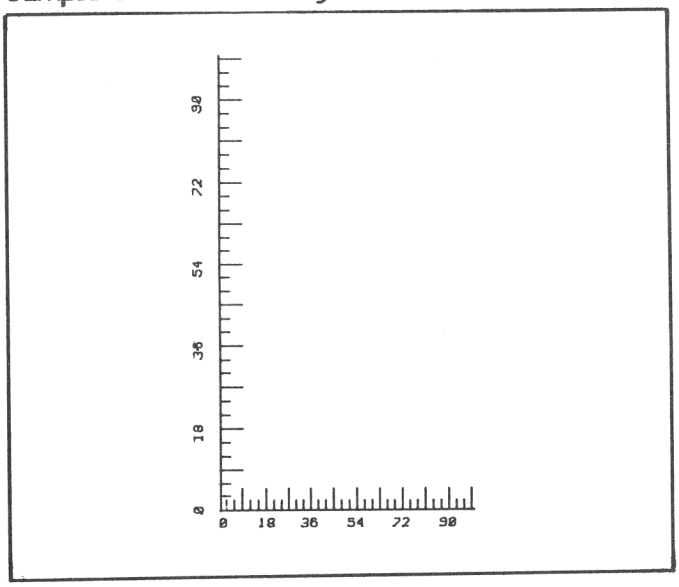
The Graphics Development module contains many subroutines which aid the user in generating graphs in X-Y coordinates, as well as two stand-alone plotting programs. The subroutines are so extensive that they cannot all be listed here. However, they can be grouped to show their general purpose as follows:

1. Graphics Initiation Subroutines. These three routines build a physical description of the graph to be drawn by later routines. How much of the plotter paper is to be used, the scale of the axes, and fixed decimal places of numeric labels are all selected in this section.

2. Plotting Initiation Section. Seven routines allow the user to describe the graph paper and draw it. The user specifies the intercept values, extreme values, axes colors, major and minor tic mark spacing, tic mark lengths, numeric label intervals and colors, label direction, grid line specifications, and framing limits. This provides almost limitless variations in producing a graph paper background for a plot. By writing a program in RAM one can call these subroutines to draw the graph paper and

then plot a scaled function on the graph. The routines are arranged in such a way that LAXIS and LGRID call all the others in this section. In LAXIS, note that the labeling as described in the manual does not work unless the axes have been drawn along the lower and left boundaries; that is, unless y(int)=min(y) and x(int)=min(x). If not, only the labels at the ends of the axes are printed. Also, the axes are drawn wherever x(int) and y(int) specify; they do not necessarily establish the origin. The main use for these routines would seem to be to aid in producing a graphing area on which to plot functions or curves. Unfortunately, regaining your coordinates (through "CURSOR") and scaling a curve to your graph is tedious work. The authors have made it very simple to produce a neat graphing area but difficult to graph on it! It is

Sample *Grid Produced by CE-501A Module.*



almost tempting to tear off the graph paper area and plot on it by hand, a somewhat self-defeating idea. Some this is saved, however, by MPLOT.

3. Plotting Programs. These are two very useful function plotting programs called MPLOT and PLOT. MPLOT is a macroplotter which offers most (but not all) of the options described in section 2 above, except that a grid pattern is always used. An error in line 20290 ignores the input color for the graph area and always selects COLOR 0. The user puts almost any number of functions to be simultaneously plotted into program memory. Unfortunately, the form of a function is "FX=" followed by an expression where the independent variable is the letter I. Thus, to plot f(x)=x*sin(x) the function is entered as FX= I * SIN(I). Confusing! But MPLOT is still impressive. A macroplotter allows boundaries for the plot to extend beyond the limits of the printer page. The graph is then produced in labeled sections which the user detaches, cuts apart, and then pastes together to form a plot of almost any size! When someone complains that the CE-150 plotter is too small to be of any use, an output of MPLOT is impressive to wave in front of them. Alternate colors are used for each function, so be sure that all pens are loaded. Plotting can be done in a continuous or point-by-point mode. The number of points plotted depends on the INCREMENT value. Note that this value should be in relatives values of x based on min(x) and max(x).

PLOT is an altogether different plotter. Several functions can be entered, as well as a few parameters, as in MPLOT. Most of the labeling and graph area is selected automatically, drawn, and then the functions are plotted in this single area. (This program is the quickest way to plot a function in this library.) Once the plot is completed, the prompt 'WINDOW?" appears. By entering the value defining a rectangular area (the window) one can selectively enlarge areas of the plot! The plot is redone, using the window as the new graph area. Again, a smaller window can be selected, enlarged, and the process continued. Since the axes are continually relabeled, one can study the behavior of a function in any neighborhood, conduct root searches, study function intersections, etc. All pens must be loaded since all are used.

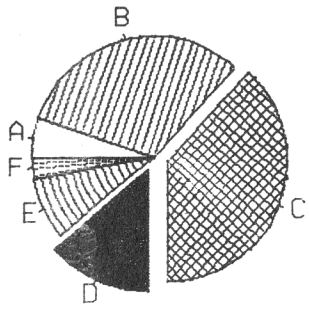
The Graphics Development Library is accompanies by of of the best manuals in the module series. The documentation of subroutines, labels, variables and parameters is extensive. Many examples are given. This module offers extensive exercises in graphics development, though its chief assets are MPLOT and PLOT.

Business Graphics (CE-501B)

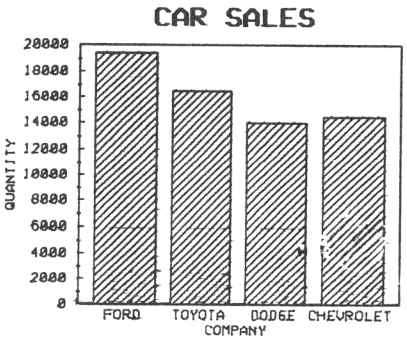
This module produces excellent business charts which use the full capabilities of the CE-150 plotter. It also includes a curve fitting program! Though the graphs produced are indeed small, they provide vual interpretations usually provided only to those with access to a graphics department. These little graphs can also be enlarged by a reproduction center. The main attribute of the Business Graphics Library is its eas of use. The programs include:

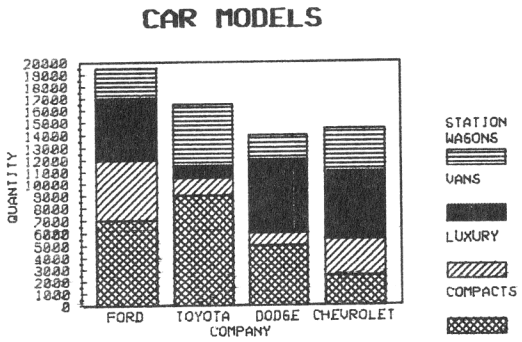
1. Pie Chart. The charts can be selected as plain (black outline, blank slices) or multicolor (allowing for several colors and shadings). The user selects the title and data for each slice, and whether or not each slice is to be "exploded"

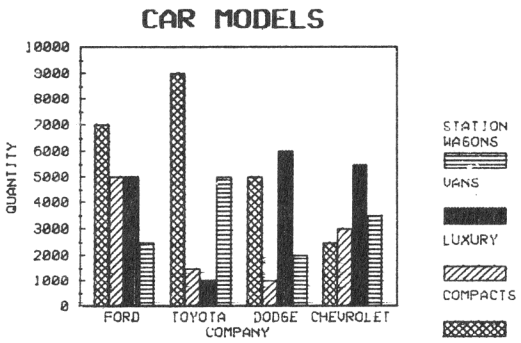
Sample *Outputs from Business Graphics (CE-501B) Module.*

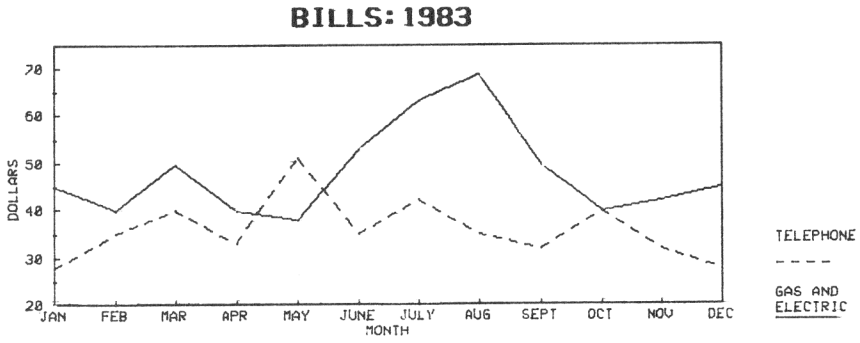


A:	ZEBRAS	20	5.27%
B:	MONKEYS	120	31.66%
C:	SNAKES	145	38.25%
D:	DEER	52	13.72%
E:	BEARS	30	7.91%
F:	LIONS	12	3.16%
TOTAL:		379	100.00%







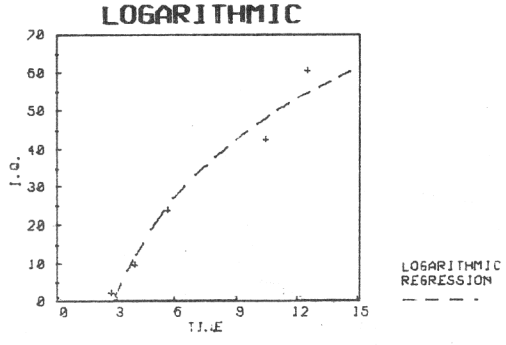


LOGARITHMIC REGRESSION

	DF	SS	MS	F
TOTAL	4	2304.2		
REG	1	2217.9	2217.9	
RESID	3	86.2	28.7	7.7E 01

R^2 = 0.9625

$\hat{Y}$ =(-3.9194E 01) +(3.7247E 01) LN(X)



(slightly offset). In multicolor, each slice can have one of four colors and one of six types of shadings. Once the choices are made, the chart is automatically drawn proportionally to the data, and a table of the data and percentages is listed.

2. Bar Chart. As in the pie charts, any bar can have any selected color and shade. The user has complete control over the titles which appear for the total graph, two axes, and each bar. Bar charts can be chosen to be normal, stacked, or grouped. Usually, selection is made based on the type of comparison of data to be made. The maximum number of data sets is 56. The maximum character length for each title in each set of x-data is 8 to insure no title overlapping. Longer titles may be used if adjacent titles are shorter. It is very impressive to simply input the titles and data for each set and to see such a precise graph be drawn so quickly!

3. Line Chart. Up to 12 lines may be plotted simultaneously in 9 different dash patterns. Up to 53 data points may be plotted per line (depending on the number of lines drawn).

4. Point Chart. The user can create a scatter chart by inputting up to 81 points using (X,Y) coordinates. Each point is plotted by an isolated dot. It is useful to make a point chart to visualize which type of curve fit best suits the data. The data may also be read from or save to a cassette file.

5. Curve Fitting. The user inputs data and can select any or all of the following analyses: linear, power, exponential or logarithmic. Associated statistics are printed and a graph of the data and fitted curve is plotted, along with the equation of the curve. Note that the x values *must* be entered in ascending order. Any x-value that is not in the proper order will be omitted. Again, cassette files may be used.

The Business Graphics Library is an excellent module for its range of offerings and its ease of use. In fact, Sharp now has on the market a "calculator/plotter" which does the exact same plots as this library, at almost the same cost as the PC-1500!

Math (CE-505A)

This module tackles some computational problems in higher mathematics and yet contains more programming errors than any other module. Numerical input allows for the entry of pi, scientifically notated, arithmetic, or function values. The programs include:

1. Prime Factors.

2. Gamma Function. Note that the input *n* must have a value less than 71. This program can be used to compute factorials.

3. Polynomial Multiplication.

4. Hyperbolic Functions. Note that the input must be less than 231. Inverse functions are also computed.

5. Relative Minimum of a Function. There is often compounded rounding error. Repeating minimal values are not detected. The endpoints of a given interval are not considered in detecting a minimal value.

6. Matrix Arithmetic including Matrix Addition, Multiplication, Inversion, Determinant, and Scalar Multiplication. The maximum allowable matrix dimension in the Determinant and Inverse programs is ten. There are also programming errors in these two programs that cause the "SINGULAR MATRIX" error to appear even when there should be no error. In the Determinant program, line 5410 announces this error if there is a zero on the diagonal from row 2 to row n of the input matrix. In the Inverse program, line 5527 detects an error if any element on the diagonal is zero in the inverse matrix as it is being constructed; this certainly occurs if the input matrix element A(1,1) is zero. These are serious flaws, as the input matrix might not necessarily be singular. A possible solution is to remove all zeros from the diagonal by row and column operations (which do not affect the determinant) in the original matrix before it is input.

7. Ordinary Differential Equations (First and Second Order). The documentation is vague in this section.

8. Roots of an Equation (f(x)=0). Newton's Method is used to approximate roots with user defined precision. The user must input f(x) and, optionally, f'(x).

9. Linear Systems of Equations. Annoyingly, the order of entry of elements of the coefficient matrix is reversed from the input of the Matrix programs. Also, if a program yields a "NOT SOLVABLE" message, one cannot retry the problem using the "EXIT(N)" option because the FOR/NEXT stack is not cleared, thereby causing an ERROR 2 in line 20045. Instead, one must exit the program and restart it. But more importantly, this program suffers from the same problems found in the Matrices programs. Upon input of a system to which a solution *does* exist, the program may signal a "NOT SOLVABLE" message! However, when the order of equations is switched, or a single equation is multiplied by a constant, the correct solution may appear. This is due, once again, to an error in line 21280 which causes the program to halt whenever a zero appears on the diagonal of the reduced augmented matrix as it is being constructed. The problem is the programmers'

algorithm for producing the reduced matrix which does not guarantee the correct ordering of the rows. The error trap is therefore incorrect. Also, their matrix reduction is halted when n-1 of the n variables are discovered; this leads to values for X(n) which do not necessarily satisfy the n'th equation! These serious flaws render this program almost useless. If the program signals "NOT SOLVABLE", this may not be the case. If it does yield a solution, this solution should be checked in all the equations.

10. Non-Linear Systems of Equations. Two non-linear systems in two unknowns are solved by Newton's approximation method. The user enters the functions, their partial derivatives, and a degree of precision. Enter the functions as the instructions on page 85 of the manual show, not as in the examples.

11. Integration. To avoid a common problem of numerical integration, investigate the asymptotic nature of the function being integrated. For example, y = 1/(x-1)^2 integrated from 0 to 2 will give an error n=5, but a numerical value when n=10, while the true value is infinite.

12. Cubic Spline Interpolation. The interpolation point x must be intermediate between x(1) and x(n). Also, the entry of the value y" at x(1) and x(n) are crucial. They should not simply be taken to zero, as in the examples.

It should be noted that the 2047 bytes of RAM are used extensively by a number of the programs. Functions are entered as program lines with confusing labels. Why couldn't a simple parsing routine have been used for function input, as Radio Shack did in its Plotter program? The manual lacks documentation as to variable usage and uses conflicting notation. The Mathematics Library often misses its mark as a useful tool.

ADDRESSING PIXELS ON THE PC-1500

This program was submitted by: *Patrick L. Pollet, University of Petroleum and Minerals, Department of Chemistry, UPM Box 298, Dhahran International Airport, P.O. Box 144, Saudi Arabia.*

one of the nice features of the PC-1500 is the possibility of producing graphics using the GPRINT instruction. Almost unlimited applications could be thought of, such as graphics games, sophisticated messages or, more seriously, redefining the keyboard by adding new characters such as French accentuated vowels or Greek or Russian alphabets.

Other than "try-and-hit" techniques, the only way to get the GPRINT hexadecimal codes for a desired pattern is to draw it on a 7 by 156 grid and then translate into hexadecimal using the "bit pattern" of every column. That process is quite long and prone to errors.

The GPRINTAID ROUTINE provided here gives you the possibility of addressing every individual dot of the screen with a flashing cursor that can be moved in any of 4 directions by pressing the following numeric keys:

"4"	cursor left by one row
"2"	cursor down by one line
"6"	cursor right by one row
"8"	cursor up by one line

Program: *GPRINTAID.*

```
3A90 58 7A 5A 00   3ADC 2C 93 29 B7   3B2C B5 40 AE 79   3B78 B7 3A 81 02
3A94 6A 9B B5 00   3AE0 0D 8B 6F B7   3B30 00 BD FF AE   3B7C B3 06 28 24
3A98 51 88 03 E9   3AE4 36 89 0F A5   3B34 79 01 91 8F   3B80 B9 0F BB 30
3A9C 78 75 00 B5   3AE8 78 75 F9 B3   3B38 9E 40 B7 35   3B84 B7 3A 81 02
3AA0 01 AE 79 00   3AEC 01 B7 9C 93   3B3C 89 07 15 AB   3B88 B3 06 2A 48
3AA4 BD FF AE 79   3AF0 3B AE 78 75   3B40 79 00 1E 9E   3B8C 7B 4A 7F B5
3AA8 01 48 74 4A   3AF4 9E 4D B7 34   3B44 3A B7 01 89   3B90 00 43 A4 0E
3AAC 00 5A 00 6A   3AF8 89 0D A5 78   3B48 07 15 A9 79   3B94 BE A7 81 B5
3AB0 9B 55 CD 88   3AFC 75 FB B1 01   3B4C 01 1E 9E 45   3B98 00 43 24 0E
3AB4 88 05 CD 8C   3B00 91 4C AE 78   3B50 9E 9C 5A 9B   3B9C BE A7 81 14
3AB8 A5 78 75 1A   3B04 75 9E 5E B7   3B54 15 89 03 56   3BA0 A7 79 00 8B
3ABC 15 AB 79 00   3B08 32 89 16 A5   3B58 9E 06 54 14   3BA4 12 A5 79 01
3AC0 FD 88 CD 88   3B0C 79 00 D9 B7   3B5C AE 79 00 BE   3BA8 B3 02 AE 79
3AC4 6A 20 BE AB   3B10 80 81 02 B5   3B60 B0 EB BE A9   3BAC 01 A7 79 F5
3AC8 2A 88 05 15   3B14 01 AE 79 00   3B64 D5 BE AB EF   3BB0 99 41 BE A9
3ACC A9 79 01 FD   3B18 BD FF AE 79   3B68 BE B7 3F 5A   3BB4 F1 9E 4A BE
3AD0 0A CD 88 6A   3B1C 01 91 76 9E   3B6C 00 E9 79 01   3BB8 B1 16 BE A9
3AD4 20 BE AB 2A   3B20 3A B7 38 89   3B70 00 55 2A F1   3BBC E4 BE AB E6
3AD8 88 05 BE E4   3B24 15 A5 79 00   3B74 B9 0F BB 30   3BC0 BA A7 69 38
                   3B28 F9 D5 81 02
```

Once the desired dot has been reached it can be turned on by pressing the number "5" key or turned off by pressing the "shift" key (for correction). In these latter two cases, the cursor will then move automatically to the next dot below or to the top of the display if the screen's limits have been crossed. This feature allows the drawing of solid black area or the erasing of an entire screen area simply by keeping the keys down. Once the desired graphic pattern has been composed, pressing the "ENTER" key will cause the printer to produce the corresponding GPRINT hexadecimal codes for archiving or other uses. (The use of any other keys except "break" is ignored.)

The routine is written entirely in machine code. This code is fully relocatable. After loading the 306 bytes in a convenient block of memory it is suggested that you protect it using the command NEW+START ADDRESS+307. You can then access the routine by using CALL ADDRESS or, even better, by using the following BASIC line:

```
1 CALL ADDRESS:WAIT:GPRINT:END
```

This line allows the pattern to stay on the screen after the displaying of the codes until you press the "ENTER" key again.

By the way, you can change the sensitivity of the keys (and the flashing rate) by modifying the value in register UL at the addresses &3AC5 and &3AD4 in the accompanying listing. (The value 20 is used at these points in the listing.)

Program *Shell Game.*

```
100:REM Shell Game
    , 22 March 198
    4
110:REM Copyright
    1984 by John R
    . Gibson
120:"S"CLEAR :DIM
    X$(0)*8,Z$(0)*
    36:RANDOM
130:X$="00000000":
    REM 4 Graphic
    Spaces
140:S$="707870C7C7C
    7C7870":REM Sh
    ell
150:L$="0E0F0F0F0F
    0F0F0E":REM Li
    fted Shell
160:P$="0E0F0F6F6F
    0F0F0E":REM Li
    fted With Pea
170:WAIT 120
180:PRINT "
     Shell Game"
190:PRINT " Cprt 1
    984, John R. G
    ibson"
200:PRINT " The PC
    -2 is quicker
    than"
210:PRINT " the ey
    e!  To move th
    e pea"
220:PRINT "  to th
    e left, use le
    ft"
230:PRINT "arrow.
     To move right
    , use"
240:PRINT "  the r
    ight arrow.  L
    ift"
250:PRINT "shells
    with up arrow.
     Set"
260:PRINT "   shel
    ls down with d
    own"
270:PRINT " arrow.
     Press R and
    the"
280:PRINT "compute
    r will hide th
    e pea"
290:PRINT "at rand
    om.  Can you g
    uess "
300:PRINT " the lo
    cation of the
    pea?":WAIT 0
310:"random"LET I=
    (RND 3)-1
320:"down"IF I=0
    THEN LET Z$(0)
    ="781414140800
    38545454180038
    44443C4000"
330:IF I=1THEN LET
    Z$(0)="7C14141
    40800385454541
    8003844443C400
    0"
340:IF I=2THEN LET
    Z$(0)="7C14141
    40800385454541
    80038444438400
    0"
350:GPRINT X$;S$;X
    $;S$;X$;S$;
360:PRINT "    Fin
    d the ";
370:GPRINT Z$(0);
380:PRINT "!"
390:"XX"IF INKEY$
    =""THEN "XX"
400:LET N$=INKEY$
410:IF N$="R"THEN
    "random"
420:IF ASC N$=8
    THEN LET I=I-1
430:IF I<0THEN LET
    I=2
440:IF ASC N$=12
    THEN LET I=I+1
450:IF I>2THEN LET
    I=0
460:IF ASC N$=11
    THEN GOTO "up"
470:GOTO "down"
480:"up"IF I<>0
    THEN "test1"
490:A$=P$:B$=L$:C$
    =L$
500:"test1"IF I<>1
    THEN "test2"
510:A$=L$:B$=P$:C$
    =L$
520:"test2"IF I<>2
    THEN "test3"
530:A$=L$:B$=L$:C$
    =P$
540:"test3"GPRINT
    X$;A$;X$;B$;X$
    ;C$;
550:PRINT "    Fin
    d the pea!"
560:"ZZ"IF INKEY$
    =""THEN "ZZ"
570:LET N$=INKEY$
580:IF N$="R"THEN
    "random"
590:IF ASC N$=8
    THEN LET I=I-1
600:IF I<0THEN LET
    I=2
610:IF ASC N$=12
    THEN LET I=I+1
620:IF I>2THEN LET
    I=0
630:IF ASC N$=10
    THEN GOTO "dow
    n"
640:GOTO "up"
STATUS 1
                1474
```

SHELL GAME

This program was submitted by *John R. Gibson, 924 Chapman Drive #9, Colorado Springs, CO 80916.* He explains it in the following manner.

Here is a simulation of the old shell game. The arrows on the PC-2 are used to move the pea and shells. Can you keep up? If you are in on the secret you will always know the location of the pea!

The Effect

You can always find the hidden pea whether it is hidden by a spectator or at random by the computer.

The Presentation

There are two ways to present the effect. First, the pea can be hidden manually so that the magician must "guess" its location. (The other players can have you leave the room or turn away while they shuffle the pea around.) Alternately, press R to have the computer randomly select a hiding place. Either way, you as the magician will not be fooled.

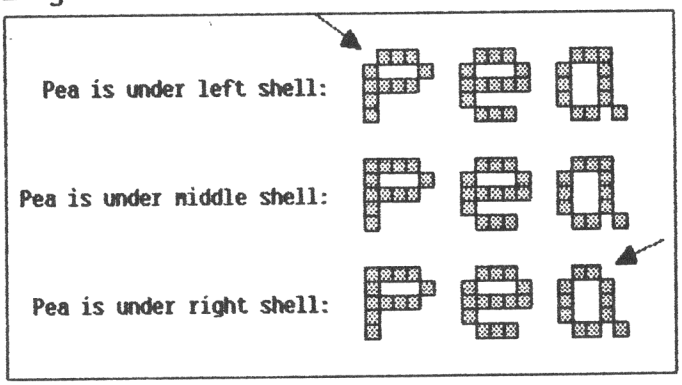
Press DEF/S to execute the program after it has been loaded. A short set of instructions is followed by a display of three shells. To lift the shells and view the pea, push the up arrow. To move the pea right use the right arrow. Use the left arrow to move the pea in that direction. After you have positioned the pea where you want it, use the down arrow to cover the pea.

Note: The program is running continuously in order to read these control keys. Thus, to stop you *must* use the break key.

The Secret

In this trick, the computer is your accomplice. It signals the location of the pea using the word "pea" in the display. If the upper left corner of the letter "p" is missing, then the pea is on the left. If the pea is in the center, no signal is given. When the pea is on the right, the upper right corner of the a is gone.

Diagram *Secret of the Shell Game!*



BINARY MAGIC

This program was submitted by *John R. Gibson, 924 Chapman Drive #9, Colorado Springs, CO 80916.* John has the following to say about his program.

This program can be enjoyed on three levels. To the computer newcomer, this program can be presented as a magic trick. It is impressive both visually and intellectually. On the second level the student will find it a practical demonstration of the binary numbering system. Play around with it for awhile and you will have a much better understanding of how this counting system works. Even advanced computerists will be amused by the program as it is unsurpassed as an example of PC graphics.

The Effect

A spectator selects a graphics object from those displayed (16 total). The person then answers four questions (truthfully) with a yes or no. The psychic computer then reveals the object the spectator had in mind.

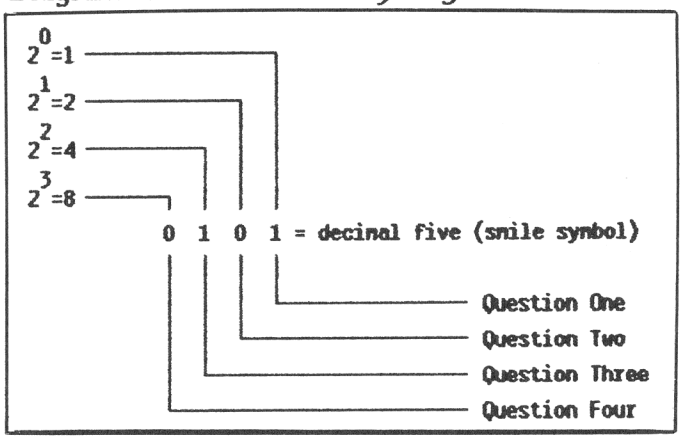
The Presentation

Load the program and hit DEF/B to execute. Some instructions will be displayed, followed by the graphics symbols. Choose one. Press the ENTER button. The computer will ask whether or not your symbol is among the ones displayed. If it is, press Y and the ENTER key. If not, then press N and the ENTER key or alternately just press ENTER. This process is repeated three more times after which the computer reveals the object that was selected by the spectator! To do the trick again, press ENTER.

The Secret

The trick is self-working. Each object is represented by a binary number from 0000 (male symbol) to 1111 (female symbol) where 0 = no and 1 = yes. See the accompanying diagram.

Diagram *The Secret of Binary Magic!*



Program *Binary Magic*

```
100:REM Binary Mag
    ic, 12 March 1
    984
110:REM Copyright
    1984 by John R
    . Gibson
120:"B"CLEAR :DIM
    A$(15)*20:
    RESTORE
130:FOR I=0TO 15
140:READ A$(I):
    NEXT I
150:WAIT 130
160:PRINT "     Bi
    nary Magic"
170:PRINT " Cprt 1
    984, John R. G
    ibson"
180:PRINT "  The P
    C-2 will guess
     the"
190:PRINT " object
     you are think
    ing"
200:PRINT " of.  C
    hoose a symbol
     from"
210:PRINT " those
    displayed.  Pr
    ess"
220:PRINT "  ente
    r key.  Answer
     "
230:PRINT "questio
    ns with a yes(
    Y) or"
240:PRINT "no (N)
    plus ENTER and
     your"
250:PRINT " symbol
     will be revea
    led!"
260:PRINT " Press
    enter to conti
    nue."
270:"G"WAIT 0:N=0
280:FOR I=0TO 14:
    GPRINT A$(I);
290:BEEP 1,I,4:
    NEXT I:WAIT
300:GPRINT A$(15):
    WAIT 0
310:GPRINT A$(1);A
    $(3);A$(5);A$(
    7);A$(9);A$(11
    );A$(13);A$(15
    );
320:INPUT "  Here(
    Y,N)?";X$
330:IF X$="Y"THEN
    LET N=N+1
340:CLS :X$=" "
350:GPRINT A$(2);A
    $(3);A$(6);A$(
    7);A$(10);A$(1
    1);A$(14);A$(1
    5);
360:INPUT "  Here(
    Y,N)?";X$
370:IF X$="Y"THEN
    LET N=N+2
380:CLS :X$=" "
390:GPRINT A$(4);A
    $(5);A$(6);A$(
    7);A$(12);A$(1
    3);A$(14);A$(1
    5);
400:INPUT "  Here(
    Y,N)?";X$
410:IF X$="Y"THEN
    LET N=N+4
420:CLS :X$=" "
430:GPRINT A$(8);A
    $(9);A$(10);A$
    (11);A$(12);A$
    (13);A$(14);A$
    (15);
440:INPUT "  Here(
    Y,N)?";X$
450:IF X$="Y"THEN
    LET N=N+8
460:CLS :X$=" "
470:FOR I=1TO 30
480:BEEP 2,250,2:
    GCURSOR 60
490:GPRINT "1C1C1C
    1C7F3E1C0800";
500:GPRINT A$(RND
    (16)-1);
510:GPRINT "00081C
    3E7F1C1C1C1C"
520:NEXT I
530:PRINT "You are
     thinking of t
    he ";:WAIT
540:GPRINT A$(N):
    CLS
550:GOTO "G"
560:DATA "00304848
    483503070000",
    "000E112142211
    10E0000","0004
    324A1429261000
    00"
570:REM Male, Hear
    t, Swirl
580:DATA "00364949
    364949360000",
    "0003474F7F4F4
    7030000","001C
    2E5B5F5B2E1C00
    00"
590:REM Clover, Go
    blet, Smilely
600:DATA "0063594F
    454F59630000",
    "00003E2222223
    E000000","0030
    28242224283000
    00"
610:REM Phone, Squ
    are, Triangle
620:DATA "007F5D7F
    777F5D7F0000",
    "001C224141412
    21C0000","0008
    1C2A7F6B7F2A1C
    08"
630:REM Die, Circl
    e, UFO
640:DATA "00003E7E
    7E7E3E24120C",
    "0040783E2B2F2
    B3E7840","0008
    142241221408000
    00"
650:REM Cup, Wumpu
    s, Diamond
660:DATA "00062979
    2906000000000":
    REM Female
STATUS 1      1724
```

INSIDE THE PC-1500

This is the third part of the current series on the Sharp PC-1500. The information is provided through the efforts of *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.* We pick up in this issue from where we stopped in *PCN* Issue 32. That is with a continuation of synopsis of a number of the routines contained in the ROM of the PC-1500. (Of course, virtually all of this information is applicable to the Radio Shack PC-2). These routines are capable of performing many useful functions when called as machine language subroutines, etc. Norlin has spent a great deal of time gleaning this information and providing an easy to read summary of their uses. The next page picks up where the previous article left off -- in the midst of presenting information on binary routines.

BINARY FORMAT IN R0

The following format is used for data in R0:

```
7A04  B2, identifying R0 as containing binary
7A05  First byte of data
7A06  Second byte of data
```

Data in this format may not be used as input for floating-point arithmetic routines. It may, however, constitute the contents of a variable; BASIC routines will correctly interpret it.

Subroutine D9E7 stores the contents of U into R0, in binary format. Following execution, A will contain UL. Y and U are unchanged.

Subroutine D9E3 stores the byte pointed to by X into R0, in binary format. Following execution, UL will contain that same byte; UH will contain 00. Y is unchanged.

CHARACTER STRING ROUTINES

STRING POINTER

A string pointer in R0 has the following format:

```
7A04  D0, identifying R0 as a string pointer
7A05  High byte, beginning address of string
7A06  Low byte, beginning address of string
7A07  Number of characters in string
```

Call 24 (DEAF) stores a string pointer into R0, taking X as the beginning address, and A as the length of the string. Y is unchanged.

Call DC (DEBC) loads X with the address, and UL and A with the length of the string pointed to by R0. Register Y is unchanged.

Subroutine D048 will copy the string beginning at X, containing UL characters, into memory beginning at address Y. After execution, Y will contain the address following that of the last character stored.

EQUIVALENTS OF BASIC FUNCTIONS

(1) String input, numeric output. Prior to execution, R0 must contain a pointer to the string; the numeric result will be placed into R0.

(a) ASC: Subroutine D9DD. Precede by loading YL with 60. Result will be in binary format.
(b) LEN: Subroutine D9DD. Precede by loading YL with 64. Result will be in binary format.
(c) VAL: Subroutine D9D7. Result will be in decimal.

(2) Numeric input, string output. Prior to execution, R0 must contain numeric data (either binary or decimal format), and 7894 must contain 10. Following execution, R0 will contain a pointer to the resulting string, and UH will contain either 00 or an error code.

(a) CHR$: Subroutine D9B1. (C1 is placed into 7A04, instead of D0)
(b) STR$: Subroutine D9CF.

STRING COMPARISONS

Subroutine D0F9 will compare the two strings indicated by string pointers in R0 and R2 (here designated as R0$ and R2$). The comparison performed will depend on the contents of A, as follows:

A=00:	R0$<>R2$	A=04:	R0$=R2$
A=01:	R0$<R2$	A=05:	R0$<=R2$
A=02:	R0$>R2$	A=06:	R0$>=R2$

If the test condition is met, R0 will contain 1 with flag Z clear; otherwise, R1 will contain 0, with flag Z set. R2 will be unchanged. (NOTE: Comparisons using A=5 or 6 will not work with ROM version A01.)

VARIABLES USED BY BASIC

LOCATING A NAMED VARIABLE

A variable specified by its name may be located by Call 0E (D461). Preceding execution, the variable sought must be specified, according to the following rules:

(1) UH must contain the ASCII code for the first character (A to Z) of the name of the variable. (@( ) and @$( ) may not be used here.)
(2) For one-character names, UL must contain zero. For two-character names, UL must contain the ASCII code for the second character--except that when the second character is 0 to 9, UL should contain 10 to 19.
(3) If a string variable is specified, UL must be increased by 20; if a dimensioned variable, by 80.
(4) When a single subscript is used, 01 must be stored into 788C, and R0 must contain the value of the subscript (in decimal).
(5) When a double subscript is used, 02 must be stored into 788C. The first subscript (in decimal) must be pushed into the BASIC data stack; the second subscript (in decimal) must be contained in R0.

Two data bytes must follow the instruction calling this subroutine. The first of these two passed bytes should be 5A. Following execution of Call 0E, U will contain the beginning address of the variable. Also, R0 will contain a variable pointer, having the following format:

```
7A04  D0
7A05  High byte of variable address
7A06  Low byte of variable address
7A07  Length of string variable; 80 if numeric
```

If a subscript is too large, or a subscripted variable was not previously dimensioned, the return address is incremented by the second passed byte, with UH containing an error code. If an non-subscripted previously-undefined variable was sought, 7A05-7A06 will contain the codes for the variable name, with 80 added to the first code.

LOADING R0 WITH VARIABLE CONTENTS

Call 0E (D461) may be used to load R0 with the contents of a variable specified by name. Preceding execution, the variable sought must be specified according to the same rules used above.

The first of two Passed bytes must be 52. Following execution, R0 will contain numeric data or a Pointer to a string, as determined by the variable specified.

If a subscript is too large, or a subscripted variable was not Previously dimensioned, the return address is incremented by the second Passed byte, with UH containing an error code. If an non-subscripted Previously-undefined variable was sought, R0 will contain zero.

CLEARING VARIABLES

Subroutine D080 clears all BASIC variables. No change in Y.

Subroutine D091 clears Main-memory variables only, by resetting the Start of Variables Pointer (7899-789A). No change in X, Y, or U.

TIME and RANDOM NUMBER

TIME (FORMAT USED BY BASIC)

Subroutine DE82 copies TIME into R0, in the same format used by BASIC. Registers XH and YH will contain 7A following execution.

Subroutine DE1D stores R0 into TIME, with R0 given in the same format used by BASIC. One data byte is Passed to this subroutine; it is added to the return address if R0 specifies TIME incorrectly.

TIME (ALTERNATE FORMAT)

Data may be transferred between R0 and the timer with the following format used in R0:

```
7A00  Not used
7A01  Not used
7A02  1st nybble=month number (hex); 2nd nybble=week day (0 to 6)
7A03  Date of month (decimal)
7A04  Hour (0 to 23, decimal)
7A05  Minute (0 to 59, decimal)
7A06  Second (0 to 59, decimal)
7A07  Not used
```

Subroutine E59A copies R0 into TIME; X and Y are unchanged.

Subroutine E5B4 copies TIME into R0; X and Y are unchanged.

RANDOM NUMBER

Subroutine F5DD Places RND (R0), interPreted as in BASIC, into R0. The random number is rePlaced by its next value. Both R2 and R6 are

used; XH and YH will contain 7A following execution.

Call 5C (F61B) replaces the random number (located 7B00-7B07, in decimal format) by its next value. Precede by loading XH and YH with 7A. Following execution, XH and YH will contain 7A. The contents of both R0 and R2 will be changed.

READING CODES FROM BASIC

EVALUATION OF EXPRESSION IN BASIC

Call DE (D6DF) evaluates an expression specified by BASIC code, beginning at the address contained in Y. The result is placed into R0. The BASIC expression is terminated by any of the codes 00 (null), 0D (ENTER), 2C (comma), 3A (colon), or 3B (semicolon).

The instruction calling this subroutine must be followed by a data byte, which will be added to the return address (with UH containing an appropriate error code) if an error occurs.

Numeric results obtained from AND, OR, NOT, ASC, LEN, POINT, PEEK, or PEEK# are in signed binary; other numeric results are in decimal. String results are expressed as string pointers in R0, with the string itself located in the String Buffer.

CONVERSION OF HEX CHARACTERS TO BYTE

Subroutine ED95 loads A with a byte determined by a pair of codes for hex digits, located starting at address X. Codes for characters other than 0-9 or A-E will clear flag C, signaling an error. After execution, X will point to the next address; Y and U are unchanged.

SEARCH FOR BASIC PROGRAM LINE

Subroutine D2EA searches for the address in memory of the BASIC line whose number (in hexadecimal) is contained in U. The search will begin at the address indicated by the START OF BASIC (or START OF BASIC in ROM) pointer. It will end if a stop byte is encountered.

The same subroutine will search for the line with a specified label. In this case, UH must contain FF, and R0 must contain a string pointer pointing to the string forming the label sought. If necessary, a search for a label will bypass stop bytes, continuing until END OF BASIC is reached.

In either case, if the line sought is located, the address of the first statement in that line is placed into 78A6-A7. (The address at which the line's number begins is less than that address by 3.)

POCKET COMPUTER NEWSLETTER
P.O. Box 232, Seymour, CT 06483